# Improvements to the Pegasus5 Overset CFD Software

Stuart E. Rogers

Computational Aerosciences Branch/Code TNA
NASA Advanced Supercomputing Division
NASA Ames Research Center, Moffett Field, CA

13th Symposium on Overset Composite Grids
and Solution Technology
October 19th, 2016

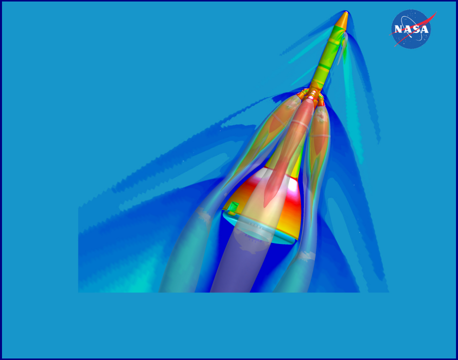
# Outline

- Introduction: motivation and background
- Improved projection routines
- Improvements to hole cutting
- Conclusion

# Introduction

## Motivation for Improvements to Pegasus5

- Complex geometries and larger grids drive need for improved automation and efficiency
  - Reduce user input
  - Reduce orphans
  - Improve hole-cutting
  - Improve parallel execution and decrease wallclock time



# Background: Pegasus5 Features and Capabilities

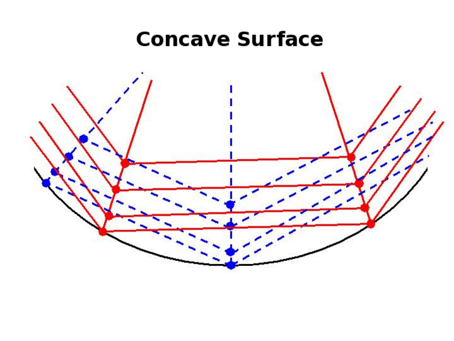
- Parallel execution using MPI
- Internal projections between overlapping surface grids
- Automatic hole-cutting
  - Multi-step hybrid method using indirect and direct hole cutting
  - Cartesian hole maps provide indirect representation of hole shape
  - Line-of-sight test using surface-grid elements: direct refined hole cutting

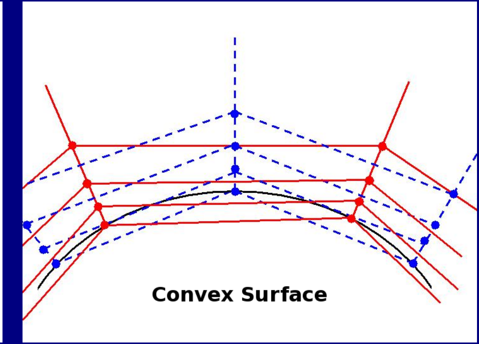
# Why Projection?

- Corrects interpolation problems that may occur on curved viscous surfaces
- Cell-aspect ratio typically $> 1000$ near viscous walls
- Pegasus5 projection step alters interpolation coefficients, not actual grid points
- Projection is performed internally and typically requires no user input

# Problem:

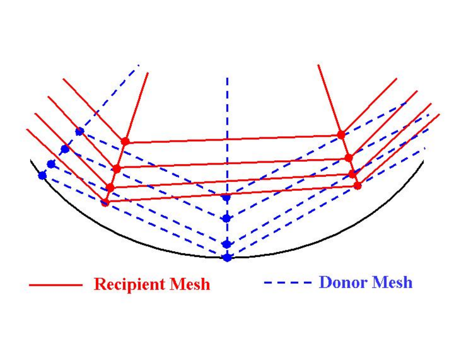
## Linear Discretization on Curved Surfaces

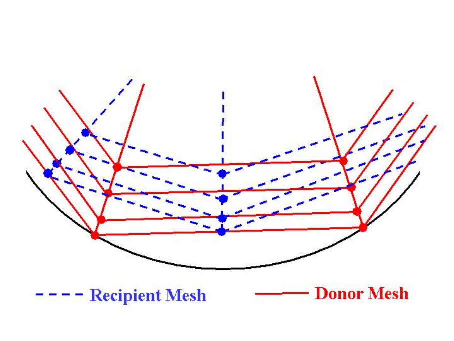




# Solution: Projection

## Points are Projected for Interpolation Only





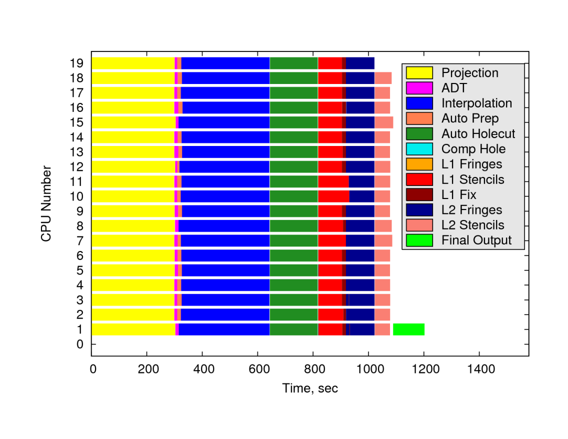
# Pegasus5 Projection Approach

- Find vector which projects a recipient's surface point onto donor's surface
- Apply filters:
  - Cannot exceed max distance
  - Cannot exceed max angle between surface normals
- Build and store list of these projection vectors
- Use for interpolation: applies projection shift to recipient grid points so that interpolation provides a stencil that is the same distance from the wall
- Actual final grid points are never moved

# Performance of Previous Projection Algorithm

## Space Launch System: 892 zones, 375 million points

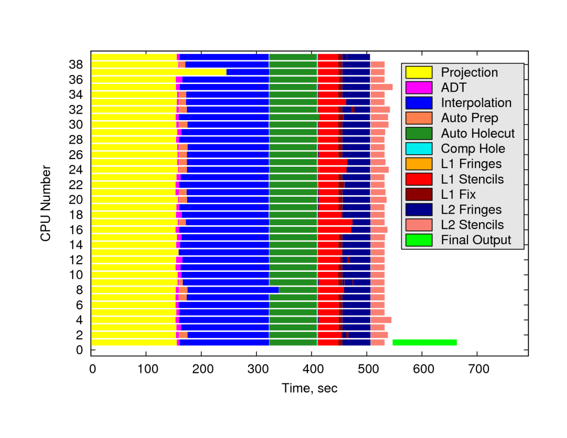
- Wallclock-time to create overset, sec:
- 20 Cores: 1205



# Performance of Previous Projection Algorithm

## Space Launch System: 892 zones, 375 million points

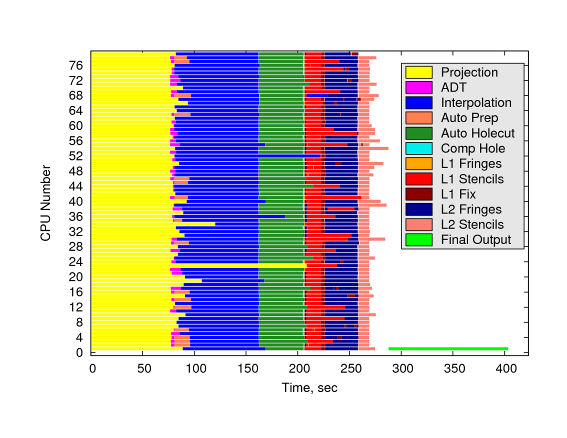
- Wallclock-time to create overset, sec:
- 20 Cores: 1205
- 40 Cores: 666



# Performance of Previous Projection Algorithm

## Space Launch System: 892 zones, 375 million points

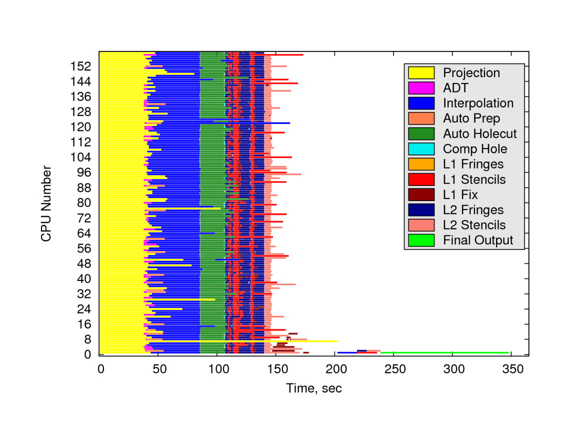
- Wallclock-time to create overset, sec:
- 20 Cores: 1205
- 40 Cores: 666
- 80 Cores: 407



# Performance of Previous Projection Algorithm

## Space Launch System: 892 zones, 375 million points

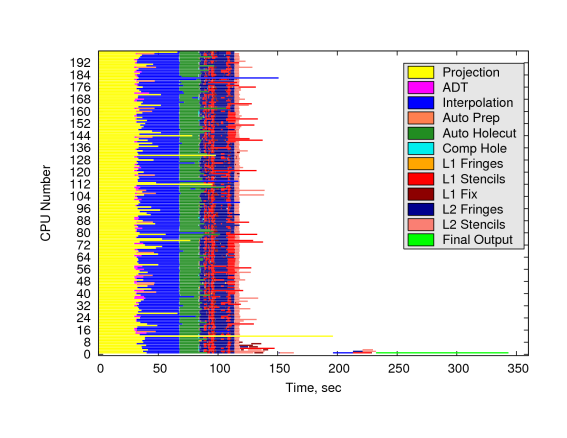
- Wallclock-time to create overset, sec:
- 20 Cores: 1205
- 40 Cores: 666
- 80 Cores: 407
- 160 Cores: 356



# Performance of Previous Projection Algorithm

Space Launch System: 892 zones, 375 million points

- Wallclock-time to create overset, sec:
- 20 Cores: 1205
- 40 Cores: 666
- 80 Cores: 407
- 160 Cores: 356
- 200 Cores: 353



Asymptotic performance: 0.94 $\mu$sec per grid-pt

Asymptotic perf excluding I/O: 0.65 $\mu$sec per grid-pt

# Improvement to Pegasus5 Projection

- Original projection process used *PROGRD* source code from the Chimera Grid Tools package
  - Volume-grid approach
  - Utilizes stencil-march search algorithm to find projection donor
  - Uses exhaustive search even for points outside donor's domain
  - Expensive approach

# New Pegasus5 Projection Approach

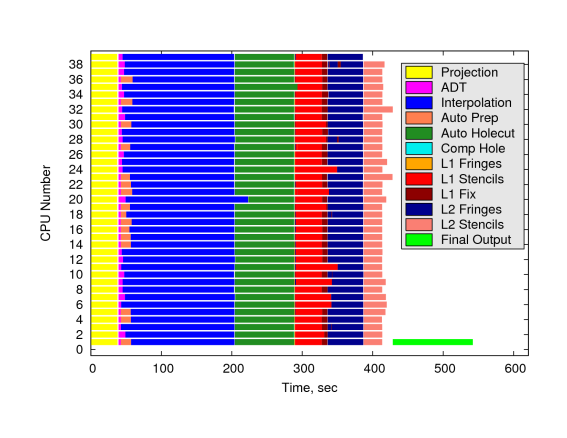
- Re-wrote entire projection process
- Use minmax box tests to rapidly eliminate most non-projecting points
- Finds surface quads closest to projection point
- Uses intersection of quad and ray through target point:
  - Bilinear surface of the reference quad
  - Ray through target point is parallel to quad's normal

- Testing verifies that:
  - New approach reproduces nearly identical results
  - New approach is 2 to 10 times faster

# Performance of New Projection Algorithm

## Space Launch System: 892 zones, 375 million points

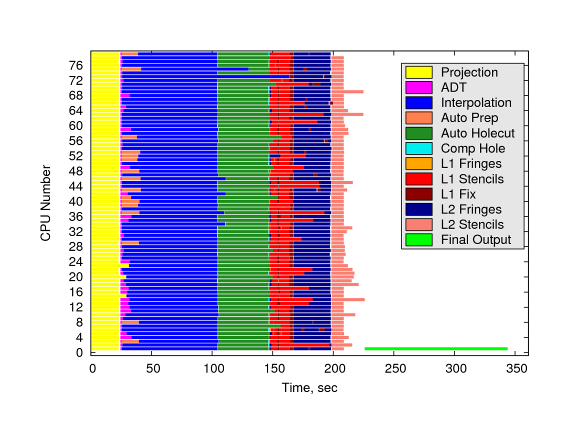
- Wallclock-time to create overset, sec:
- 40 Cores: 544



# Performance of New Projection Algorithm

Space Launch System: 892 zones, 375 million points

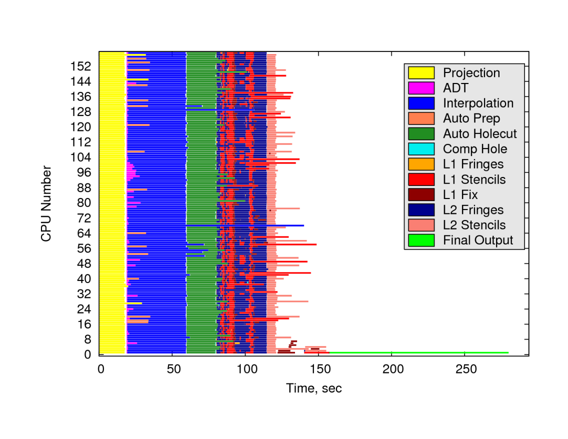
- Wallclock-time to create overset, sec:
- 40 Cores: 544
- 80 Cores: 349



# Performance of New Projection Algorithm

## Space Launch System: 892 zones, 375 million points

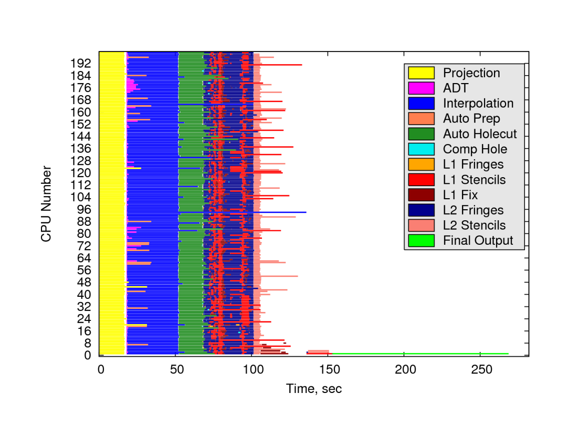
- Wallclock-time to create overset, sec:
- 40 Cores: 544
- 80 Cores: 349
- 160 Cores: 285



# Performance of New Projection Algorithm

Space Launch System: 892 zones, 375 million points

- Wallclock-time to create overset, sec:
- 40 Cores: 544
- 80 Cores: 349
- 160 Cores: 285
- 200 Cores: 277



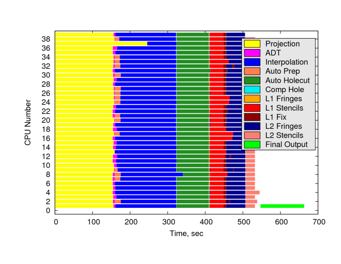
Asymptotic performance: 0.74 $\mu$sec per grid-pt

Asymptotic perf excluding I/O: 0.43 $\mu$sec per grid-pt

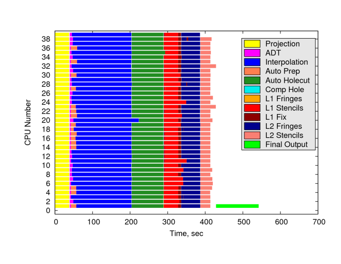
# Performance of Old Vs New Projection

## 40 Processors

OLD



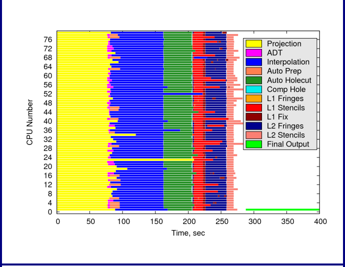
NEW



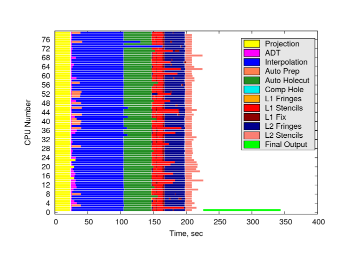
# Performance of Old Vs New Projection

## 80 Processors

OLD



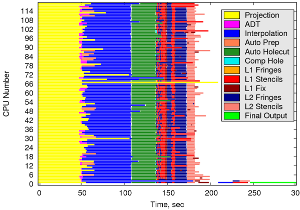
NEW



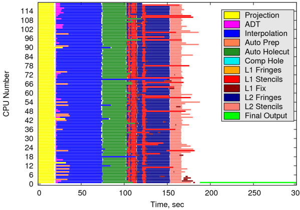
# Performance of Old Vs New Projection

## 120 Processors

OLD



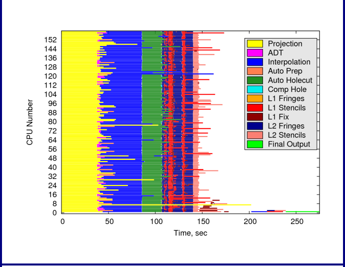
NEW



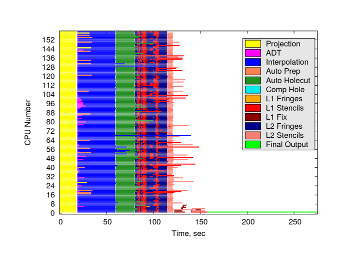
# Performance of Old Vs New Projection

## 160 Processors

OLD



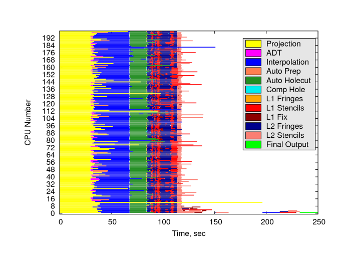
NEW



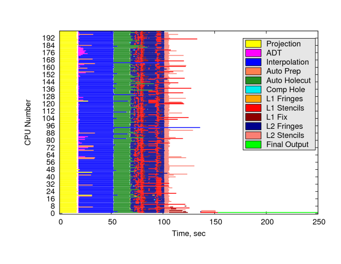
# Performance of Old Vs New Projection

## 200 Processors

OLD



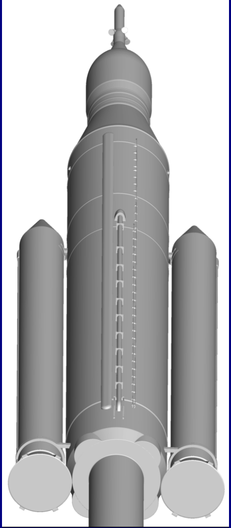
NEW



# Hole-Cutting Challenges: Protuberances

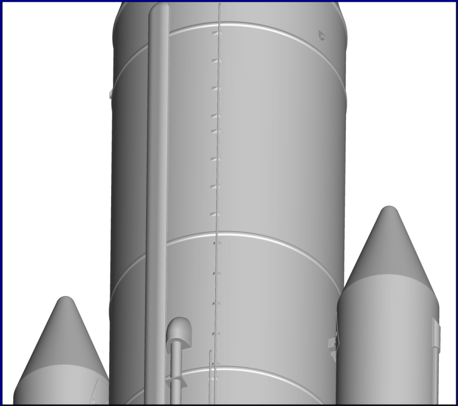
- Automatic hole cutting can handle many complex geometries
- Small protuberances: large disparity in length scales

- Example: Space Launch System wind-tunnel model
- Protuberance: core camera



# Hole-Cutting Challenges: Protuberances

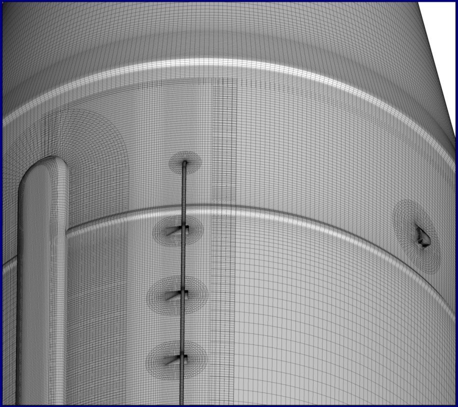
- Automatic hole cutting can handle many complex geometries
- Small protuberances: large disparity in length scales
- Example: Space Launch System wind-tunnel model
- Protuberance: core camera



# Hole-Cutting Challenges: Protuberances

- Automatic hole cutting can handle many complex geometries
- Small protuberances: large disparity in length scales

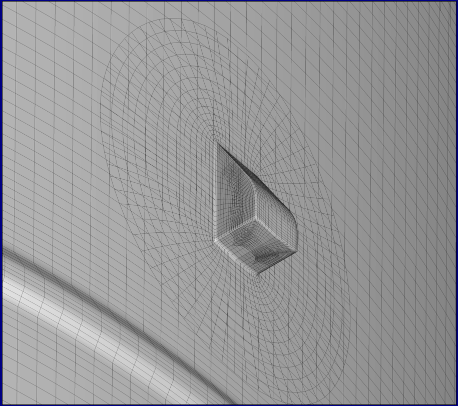
- Example: Space Launch System wind-tunnel model
- Protuberance: core camera



# Hole-Cutting Challenges: Protuberances

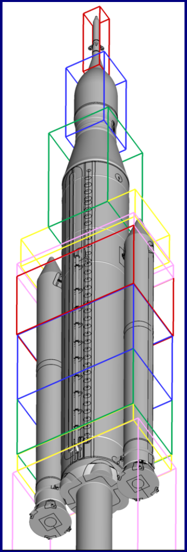
- Automatic hole cutting can handle many complex geometries
- Small protuberances: large disparity in length scales

- Example: Space Launch System wind-tunnel model
- Protuberance: core camera



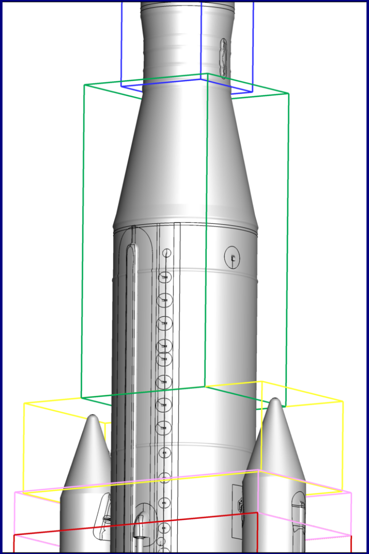
# Hole-Cutting Challenges: Protuberances

- Automatic creation of hole cutters: `AUTOHCT=10`



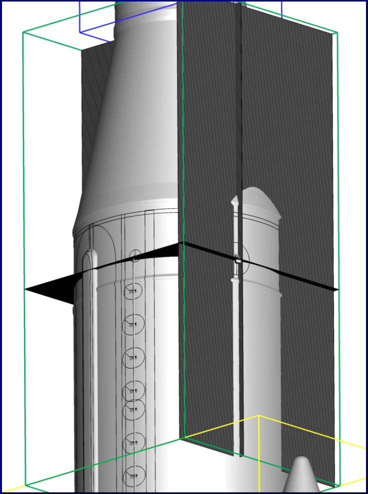
# Hole-Cutting Challenges: Protuberances

- Automatic creation of hole cutters: `AUTOHCT=10`
- Cartesian hole maps resolve space around vehicle



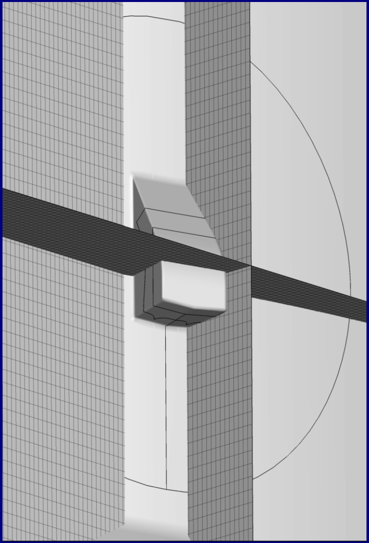
# Hole-Cutting Challenges: Protuberances

- Automatic creation of hole cutters: `AUTOHCT=10`
- Cartesian hole maps resolve space around vehicle



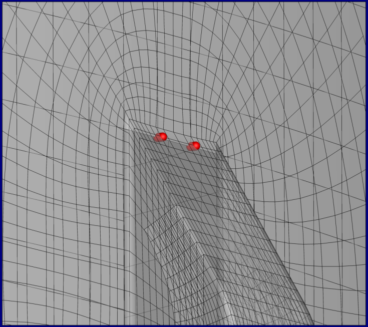
# Hole-Cutting Challenges: Protuberances

- Automatic creation of hole cutters: `AUTOHCT=10`
- Cartesian hole maps resolve space around vehicle
- Small protuberances require additional hole-cutter resolution



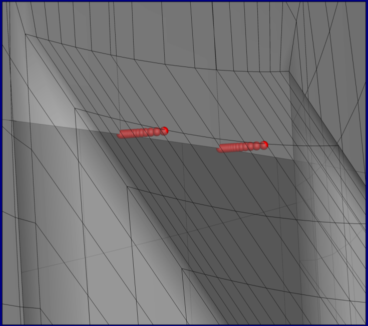
# Hole-Cutting Challenges: Protuberances

- Automatic creation of hole cutters: `AUTOHCT=10`
- Cartesian hole maps resolve space around vehicle
- Small protuberances require additional hole-cutter resolution
- Orphans: 48 grid points remain inside the protuberance



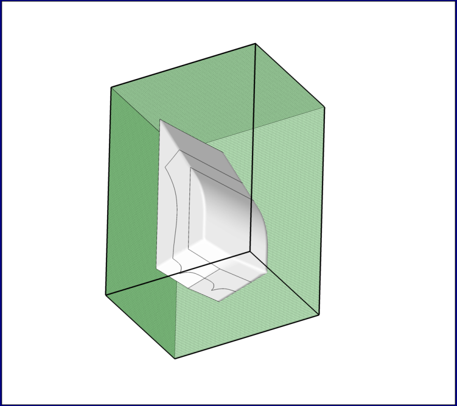
# Hole-Cutting Challenges: Protuberances

- Automatic creation of hole cutters: `AUTOHCT=10`
- Cartesian hole maps resolve space around vehicle
- Small protuberances require additional hole-cutter resolution
- Orphans: 48 grid points remain inside the protuberance



# Hole-Cutting Improvements: Protuberances

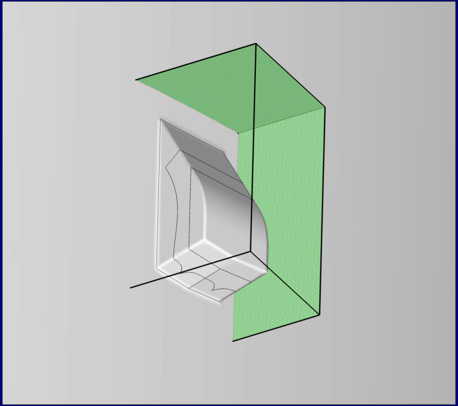
- Add a custom hole cutter using HCUT namelist
- Specify the minmax box surrounding the protuberance
- Need water-tight boundaries for flood-fill painting to work



# Hole-Cutting Improvements: Protuberances

- Add a custom hole cutter using HCUT namelist
- Specify the minmax box surrounding the protuberance
- Need water-tight boundaries for flood-fill painting to work

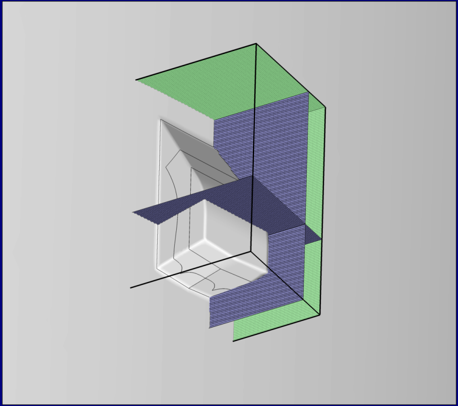
- By default: painting marks all eight corners as “Outside”
- New `HCUT` inputs: `OCORNER` controls painting algorithm
- Limit “Outside” corners to the four outside the domain



# Hole-Cutting Improvements: Protuberances

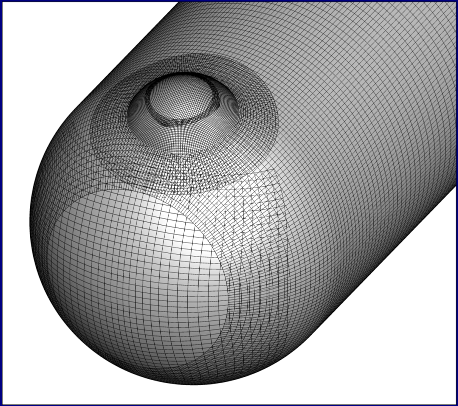
- Add a custom hole cutter using HCUT namelist
- Specify the minmax box surrounding the protuberance
- Need water-tight boundaries for flood-fill painting to work

- By default: painting marks all eight corners as "Outside"
- New `HCUT` inputs: `OCORNER` controls painting algorithm
- Limit "Outside" corners to the four outside the domain
- Zero orphans



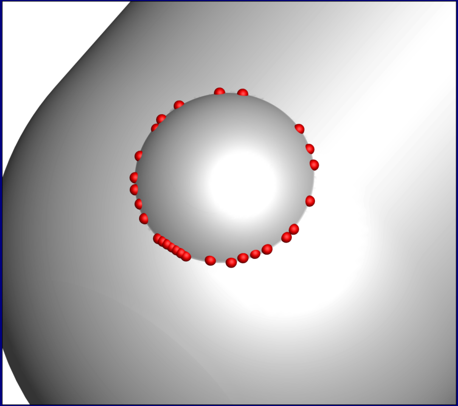
# Improvements to Hole-Cutting Process

- Test case: bump on a cylinder



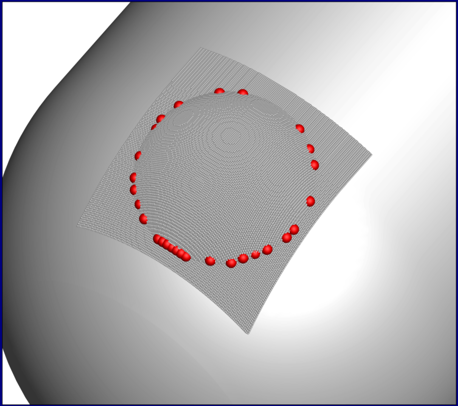
# Improvements to Hole-Cutting Process

- Test case: bump on a cylinder
- 520 orphans inside the bump



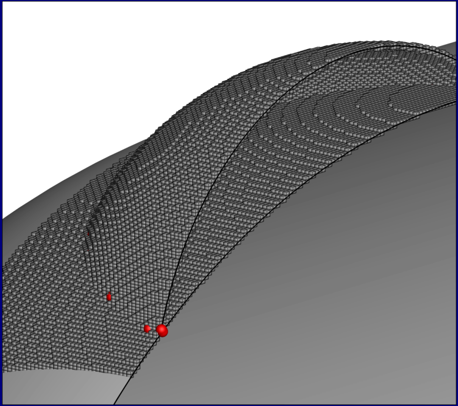
# Improvements to Hole-Cutting Process

- Test case: bump on a cylinder
- 520 orphans inside the bump
- Use HCUT hole cutter surrounding bump 128x128x128



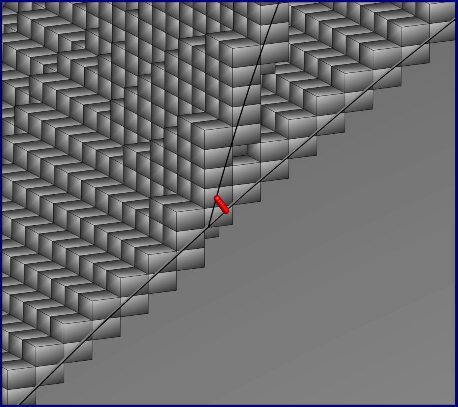
# Improvements to Hole-Cutting Process

- Test case: bump on a cylinder
- 520 orphans inside the bump
- Use `HCUT` hole cutter surrounding bump 128x128x128
- *Fringe* elements: those intersecting surfaces



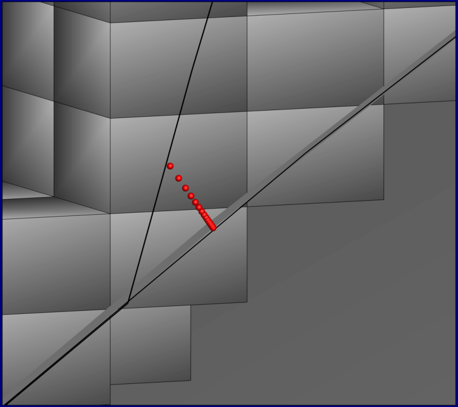
# Improvements to Hole-Cutting Process

- Test case: bump on a cylinder
- 520 orphans inside the bump
- Use HCUT hole cutter surrounding bump 128x128x128
- *Fringe* elements: those intersecting surfaces
- No line-of-sight for some points inside the bump



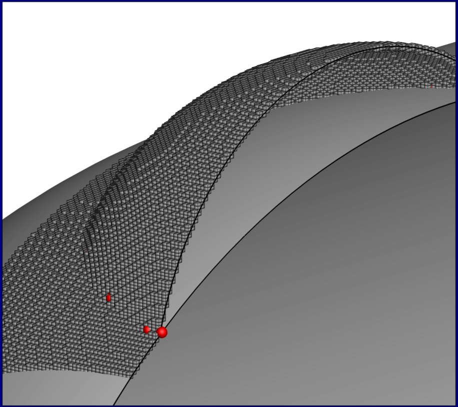
# Improvements to Hole-Cutting Process

- Test case: bump on a cylinder
- 520 orphans inside the bump
- Use HCUT hole cutter surrounding bump 128x128x128
- *Fringe* elements: those intersecting surfaces
- No line-of-sight for some points inside the bump



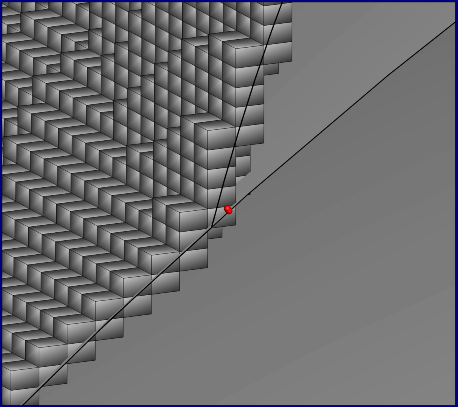
# Improvements to Hole-Cutting Process: Work in Progress

- Use additional pass in painting process:
- Mark *Fringe* elements as *Inside* elements if they are surrounded by *Inside* elements



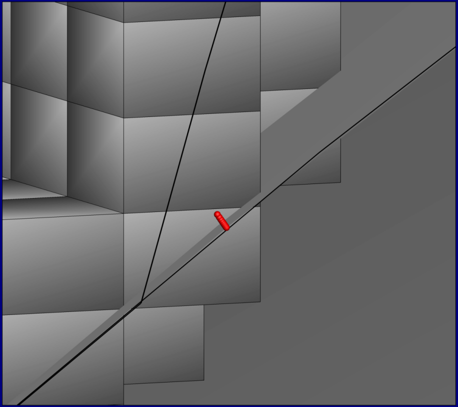
# Improvements to Hole-Cutting Process: Work in Progress

- Use additional pass in painting process:
- Mark *Fringe* elements as *Inside* elements if they are surrounded by *Inside* elements
- Reduces number of orphans



# Improvements to Hole-Cutting Process: Work in Progress

- Use additional pass in painting process:
- Mark *Fringe* elements as *Inside* elements if they are surrounded by *Inside* elements
- Reduces number of orphans
- Some orphans remain: no clear line-of-sight to *Inside* element
- Next step: maybe remove blanked surface and retry line-of-sight test



# Conclusion

- New projection routines:
  - Removes big bottle-neck
  - Improves parallel performance
- Additional inputs to control flood-fill painting enables individual `HCUT` hole cutters for small features
- Released version 5.2b of Pegasus
- Working on potential improvements to hole-cutting
- Future Work:
  - More improvements to hole cutting
  - Changes to enable Overflow mesh adaptation with Pegasus5 grids